Upper and lower bounding techniques for frequency assignment problems

Upper and lower bounding techniques for frequency assignment problems

C.A.J. Hurkens
S.R. Tiourine *

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands
e-mail: sergeyt@win.tue.nl

July 17, 1995

Abstract

We consider two variants of the radio link frequency assignment problem. These problems arise in practice when a network of radio links has to be established. Each radio link has to be assigned a particular frequency. The interference level between the frequencies assigned to the different links has to be acceptable, since otherwise communication will be distorted. The frequency assignments have to comply with certain regulations and physical characteristics of the transmitters. Moreover, the number of frequencies is to be minimized, because each frequency used in the network has to be reserved at a certain cost.

We develop several approximation algorithms for the problems, which are based on local search, and we compare their performance on some practical instances. Lower bounding techniques based on nonlinear programming and the chromatic number of a graph are used to estimate the quality of the approximate solutions for these instances.

*Partially supported by the EUCLID project RTP 6.4 as part of CEPA 6 ("Artificial Intelligence")

1 Problem description

The subject of this paper is a variant of the radio link frequency assignment problem (RLFAP). In this problem, we are given a set L of links. For each link i a frequency f_i has to be chosen from a given domain D_i. Some links may already have a preassigned frequency p_i, which may or may not be changed. Restrictions are defined on pairs of links that limit the choice of frequencies for these pairs. For a given pair of links $\{i, j\}$ these restrictions are either of type

$$|f_i - f_j| > d_{ij} \tag{1}$$

or of type

$$|f_i - f_j| = d_{ij} \tag{2}$$

for a given distance $d_{ij} \geq 0$. Two links i and j involved in a constraint of type (1) are called *interfering* links, and the corresponding d_{ij} is the interfering distance. Two links bound by a constraint of type (2) are referred to as a pair of *parallel* links; every such link belongs to exactly one such pair.

Some of these restrictions may be violated at a certain cost. Such restrictions are called *soft*, in contrast to the *hard* constraints, which may never be violated. The constraints of type (2) are always hard, and so are some of the preassignment constraints. *Interference costs* C_{ij} for violating soft constraints of type (1) and *mobility costs* M_i for changing soft preassigned frequencies are given. An assignment of frequencies is *complete* if every link in L has a frequency assigned to it. We denote by W and U the sets of all soft interference and mobility constraints, respectively.

The *first order problem* is to find a complete assignment that satisfies all *hard* constraints and is of minimum cost:

$$\text{minimize} \sum_{W} C_{ij}\delta(|f_i - f_j| \leq d_{ij}) + \sum_{U} M_i\delta(|f_i - p_i| > 0) \tag{3}$$

subject to the hard constraints:

$$\begin{array}{rcll} |f_i - f_j| & > & d_{ij} & \text{for all pairs of links } \{i,j\} \text{ involved in the hard constraints,} \\ |f_i - f_j| & = & d_{ij} & \text{for all pairs of parallel links } \{i,j\}, \\ f_i & = & p_i & \text{for all links } i \in L \text{ with hard preassigned frequencies,} \\ f_i & \in & D_i & \text{for all links } i \in L, \end{array}$$

where $\delta(.)$ is 1 if the condition within brackets is true and 0 otherwise.

If there exists a *feasible* assignment, i.e., a complete assignment of zero cost, then the *second order problem* is to find a feasible assignment that satisfies *all* constraints and minimizes the number of distinct frequencies used:

$$\text{minimize } |\cup_i \{f_i\}| \tag{4}$$

subject to the hard and soft constraints:

$$\begin{array}{rcll} |f_i - f_j| & > & d_{ij} & \text{for all pairs } \{i,j\} \text{ involved in the soft and hard constraints,} \\ |f_i - f_j| & = & d_{ij} & \text{for all pairs of parallel links } \{i,j\}, \\ f_i & = & p_i & \text{for all links } i \in L \text{ with preassigned frequencies,} \\ f_i & \in & D_i & \text{for all links } i \in L. \end{array}$$

Sometimes another version of the second order problem is considered, where the largest frequency used instead of the number of frequencies is minimized.

The paper is organized as follows. Section 2 sketches the approach we used to derive approximate solutions. The detailed description of the approximation algorithms is given in sections 3 and 4. In section 5 lower bounding techniques are described. Sections 6 presents the results obtained for the test instances and in section 7 conclusions of this study are listed.

2 Solution approach

The main aim of this work is to develop approximation algorithms for the RLFAP and compare their relative performance on real-life instances provided by the CELAR (Centre d'ÉLectronique de l'ARmement) in the context of the CALMA (Combinatorial ALgorithms for Military Applications) project. A wide variety of approximation and optimization techniques has been tested within this project. The algorithms we considered are all based on the principle of local search, which is a powerful tool often used for solving hard combinatorial problems. The general idea of local search is to start with an initial solution and iteratively perform small transformations of this solution in an attempt to improve it with respect to a given criterion.

The *neighborhood* of a given solution is defined as the set of solutions to which a given one can be transformed in one iteration. A mapping that specifies a neighborhood of each solution is called a *neighborhood function*. A solution that has no solution with better objective value in its neighborhood is called a *local optimum*. There are several neighborhood search strategies for finding good solutions.

The basic *iterative improvement* method modifies at each step a given solution to a solution from its neighborhood of lower cost. This method stops when no better solution can be found in the neighborhood.

Simulated annealing [1] modifies a solution to a random one chosen from its neighborhood. Improvements are always accepted. Deteriorations are accepted with a certain probability, decreasing during the run. Simulated annealing stops when this probability becomes smaller than a specified parameter.

Taboo search [4],[5] always performs a move to the best solution in a neighborhood. In this way the cost of solutions visited during the search is not necessarily always decreasing. To prevent this method from cycling, several recently visited solutions are removed from

any neighborhood. A stopping criterion has to be defined, for example the maximum number of iterations without improvement.

Variable-depth search [8] performs a sequence of changes at each iteration. This sequence of changes is terminated if an improvement is found or if no gain is expected from further changes. The search will be continued with a new solution in the former case or with the old solution in the latter case. The stopping criterion can be defined in the same way as for taboo search.

In order to apply these techniques to the RLFAP, we consider a natural partition of the problem into two phases, corresponding to the two different objectives. Splitting the problem in this way helps to exploit the underlying special structure. Clearly, both of these problems are NP-hard and thus no polynomial-time optimization algorithm is likely to exist [3].

A solution to a given minimization problem obtained by an approximation algorithm provides an upper bound for the optimal solution to this problem. In order to estimate the difference between the upper bounds and the optimum we also obtain some lower bounds for the test instances.

We define the notion of an *interference graph*, which will be helpful to illustrate our arguments. The set of links constitutes the set of nodes of the interference graph. If two links are bound by a constraint, then there is an edge between the corresponding nodes.

3 The first order problem

3.1 Neighborhood

We define the cost of the link i in solution S to be the sum of its mobility cost (if the frequency of i in S is not p_i) and half the sum of the penalties for all violated soft constraints involving i. The objective function in this problem is the sum of the contributions of individual links. In some cases it is possible to change a frequency of one separate link so as to reduce its contribution and thus to reduce the total cost. We define the neighborhood of a given solution as the set of all solutions that can be obtained by changing a frequency of only one pair of parallel links. At each step, however, we only consider those links whose contribution is strictly positive; we denote the corresponding neighborhood function by **R**. A neighborhood function is called *connected* if from every starting solution an optimal solution can be reached using this function.

Lemma 1 *The neighborhood function* **R** *described above is connected.*

Proof Consider an arbitrary starting solution S and let S_{opt} be some optimal solution. If the objective values of S and S_{opt} are equal, then S is also optimal. Otherwise, as long as there exists a link i of strictly positive cost in S whose frequency is different in S and S_{opt}, modify S by assigning to i its frequency in S_{opt}. Now suppose that no such link exists in S. Then all the links of strictly positive cost in S have the same frequency in S and S_{opt}. By the definition of the cost all such links have the same cost in S and S_{opt}. Hence,

the sum of costs over all links in S, which is the total cost of S, cannot be larger than the cost of S_{opt}. Thus starting with an arbitrary solution we construct an optimal solution using only transitions to neighboring solutions, which proves the lemma. □

As an example of the advantage of this reduced neighborhood consider an instance where the underlying interference graph consists of several disconnected components. Then this instance can be decomposed into several independent ones. We argue that to some extent this is done automatically by the reduced neighborhood. If during the search a frequency assignment of zero cost is found for some component, this component will not be disturbed again, which will reduce the size of the instance. This will not be the case if changes were also allowed at the links of zero cost.

We found it convenient to relax the hard constraints (4) and to introduce the corresponding term into the objective function. The penalty we attach to the violation of these constraints should be relatively high, for example the sum of weights of all soft constraints plus 1. Then from the value of the objective function one can judge whether the solution is feasible. At the same time, constraints (5) and (6) are always treated as hard.

All methods described in this section use the same procedure to obtain a starting solution. Initially a random solution is generated by assigning to each pair of parallel links a pair of frequencies randomly drawn from their domains; this solution is then subjected to standard iterative improvement, using neighborhood **R**. Each of the following algorithms has its own way of trying to escape from a local optimum.

3.2 Simulated annealing

Simulated annealing starts with some initial solution and performs a transformation to a random member of its neighborhood. We define this transformation in the following way:

- Select randomly a pair of parallel links of nonzero cost.
- Assign to them a random pair of frequencies from their domains, but distinct from the currently assigned one.

The result of the transformation is accepted as a new solution with a certain probability p; otherwise the previous solution is restored. In order to define p, we first describe some auxiliary control parameter T. Initially T is equal to the highest cost coefficient of the instance. Every k iterations T decreases according to the rule

$$T := \frac{T}{1 + [T \ln(1+\delta)/3\sigma)]} ,$$

where δ is a parameter that controls the rate of decrement of T and σ is an estimate of the standard deviation of the cost value, calculated for solutions visited in the previous k iterations. The values of δ and k have to be specified. The estimate of the standard

deviation is calculated by

$$\sigma^2 = \frac{k \sum_{i=1...k} Y_i^2 - (\sum_{i=1...k} Y_i)^2}{k(k-1)},$$

where Y_i $(i = 1, ..., k)$ is the objective value during the i-th iteration. If we denote the cost of old and new solutions by C_o and C_n, then p is defined as

$$p = \min\{1, \exp(-(C_n - C_o)/T)\}.$$

These transformations are performed repeatedly as long as T is sufficiently large. Note that a new solution of cost $C_n \leq C_o$ is always accepted. Deteriorations may also be accepted, but this will happen with decreasing probability.

3.3 Taboo search

Taboo search starts with an initial solution generated as described in Section 3.1. It performs a move to a solution selected from a given neighborhood. We used a first-improvement strategy in our implementation of taboo search for the first order problem. This strategy always moves to the first encountered nontaboo solution that yields an improvement. If no such solution is found, then it selects the best nontaboo solution from the neighborhood.

To prevent the algorithm from cycling we forbid several recently performed moves to be repeated. For this purpose we maintain a taboo list of all links whose frequency was modified in the previous k iterations; parameter k has to be specified. We arrived at this choice after experimenting with several alternatives for the items to be put on the taboo list. We also used a so-called aspiration criterion to override the forbidden status of a solution in case it would improve upon the best solution found so far.

3.4 Variable-depth search

The method for generating an initial solution produces a locally optimal solution. In each iteration we apply a combination of variable-depth search and iterative improvement in an attempt to find improvements. In our implementation of variable-depth search we reassign frequencies to the nodes of some connected component of the interference graph. We start by randomly reassigning frequencies to a pair of parallel links of nonzero cost. Note that the objective function cannot be improved by this change, because the starting solution is always a local optimum. We apply a random search to the links involved in violated soft constraints in attempt to obtain an improvement. This operation can result in a significant modification of the solution. The solution obtained in this way need not be locally optimal anymore, so we apply iterative improvement each time we find an improvement by variable-depth search.

Suppose we assign a pair of frequencies $\{f, g\}$ to a pair of parallel links $\{i, j\}$ in some given solution, while keeping the rest of the solution unchanged. Denote then by I_{fg}^{ij} the

set of all links adjacent to i or j in the interference graph whose frequency violates some soft constraint involving i or j. We now describe our variable-depth search procedure:

1. Select randomly a pair of parallel links $\{k, l\}$ of nonzero cost. Let $T = \{k, l\}$.
2. Assign a random pair of frequencies $\{f, g\}$ to $\{k, l\}$ from their domain, distinct from the currently assigned one. Let C be equal to the resulting cost increment and $S = I_{fg}^{kl}$.
3. Select randomly link i from S and its parallel link j. Let $S := S \backslash \{i, j\}$ and add $\{i, j\}$ to T. Assign to this pair of links $\{i, j\}$ a pair of frequencies $\{f, g\}$ with the lowest cost from their domains. Add the resulting cost increment to C. If the reassignment of frequencies to i and j resulted in an improvement, add the set $I_{fg}^{ij} \backslash T$ to S.
4. If S is empty or C is negative, then stop, otherwise return to step 3.

If no improvement is found in a fixed number of iterations N, we modify the current solution randomly. For that we select links with a cost contribution higher than the average and reassign a frequency to them with a given probability p. The solution obtained in this way is subjected to iterative improvement and variable-depth search, in this order. The number of times that random modification is applied is limited by a control parameter M, which also serves as a parameter in the stopping criterion. Thus, there are three control parameters: N, M and p.

4 The second order problem

We define a very different kind of neighborhood for the second order problem. We are convinced that a 'wild' neighborhood that eliminates a particular frequency from an assignment altogether is more effective in this problem than any neighborhood based on the small changes. Let F be the set of all frequencies used in some assignment. We will select a frequency f from F and try to obtain a feasible complete assignment using frequencies from $F \backslash \{f\}$. If this is not possible, then any frequency but f is allowed to be used to complete the assignment. The precise description of this mechanism will be given in section 4.2.

We use a feasible complete assignment with respect to the second order problem to initiate all approximation algorithms of this section. In some cases such a solution can be obtained relatively easily. Such cases can be treated by a greedy heuristic, described in the following section, which tries to extend a partial assignment at each step. If this heuristic fails to produce a complete assignment, then one of the methods for the first order problem can be used to obtain a solution of zero cost. The methods for the first order problem can also be used when feasibility is lost during the search for the second order problem.

4.1 Constructing a starting solution

There are two main strategies known from the literature (see for example [7]) to construct a feasible partial solution to the RLFAP.

The *frequency exhaustive* strategy orders links and assigns frequencies to them in this order. The ordering of links should reflect the potential difficulty of assigning a frequency to them. Thus the difficult links are treated first.

In the *requirement exhaustive* strategy frequencies are assigned one by one in some order. At each step a frequency is assigned to as many links as possible. This strategy takes advantage of assigning at each step a frequency to some large independent set of links.

We describe here a heuristic of the frequency exhaustive type. An initial ordering π of the set of links L is determined as follows:

1. Let S be a set initialized to L.
2. Select a link with the smallest degree in the subgraph of the interference graph induced on S. In case of ties, select a link i with the smallest total interference distance $\sum_{j \in S \setminus \{i\}} d_{ij}$ in this subgraph.
3. Remove link i from S.
4. If S is not empty, then return to step 2, otherwise stop.

The initial ordering π will be the reverse of the order of removal.

Now suppose that some partial assignment has been made. Let F_i be the set of frequencies available at link i in this assignment. This set is computed by removing from domain D_i all frequencies eliminated by the interference with links that have already been assigned a frequency. Algorithm A selects at each step a pair of parallel links and assigns a pair of frequencies to them. For the assignment of frequencies to a pair of parallel links $\{i, j\}$, all pairs of frequencies $\{f_i, f_j\}$ such that $f_i \in F_i$,$f_j \in F_j$ and $|f_i - f_j| = d_{ij}$ are checked. Among these pairs the one most heavily occupied in the current partial assignment is chosen. If none of them is used, a smallest pair is taken.

Algorithm A

1. Determine the initial ordering π. Let S be equal to L.
2. Select a link $i \in S$ with the smallest number of available frequencies:

$$i = \arg\min_{k \in S} |F_k|.$$

   In case of ties the first link in the ordering π is chosen.
3. Assign a pair of frequencies $\{f_i, f_j\}$ to a pair of parallel links $\{i, j\}$, as described above, if possible. Remove i and j from S.
4. If S is not empty, then return to step 2, otherwise stop.

4.2 Neighborhood

Suppose we assign a pair of frequencies $\{f, g\}$ to a pair of parallel links $\{i, j\}$ in some given solution, while keeping the rest of the solution unchanged. Denote by I_{fg}^{ij} the set of all links adjacent to i or j in the interference graph whose frequency violates some constraint involving i or j.

For link i, let the set F_i of alternative frequencies be as defined above, and let U_i be the set of frequencies from domain D_i used in the current solution. Denote by M_i the following subset of U_i:

$$M_i = \{f \in U_i | \exists g \in D_j : |f - g| = d_{ij} \wedge [I_{fg}^{ij} = \emptyset \vee \min_{l \in I_{fg}^{ij}} |F_l| > 1]\}.$$

The set M_i can be interpreted as a set of feasible alternatives in domain D_i. We now define the neighborhood of a given solution S. Solution S_1 is a neighbor of solution S if S_1 can be obtained from S as a result of the following procedure:

1. Choose some *seed* frequency f used in solution S (possibly a dummy frequency). Prohibit the use of f for the assignment.

2. Determine the set P of all unassigned links and links with the chosen frequency f in solution S; in case of a dummy f, P contains only unassigned links.

3. For each link i from P and its parallel link j, find a pair of frequencies $\{g_i, g_j\}$, $g_i \neq f$, $g_j \neq f$, $|g_i - g_j| = d_{ij}$ such that

   $$\{g_i, g_j\} = \arg \min_{h \in M_i, k \in M_j} |I_{hk}^{ij}|;$$

   if ties occur, choose a pair $\{g_i, g_j\}$ that maximizes the minimum number of alternative frequencies at interfering links

   $$\min_{l \in I_{g_i g_j}^{ij}} |F_l|.$$

   If such a pair of frequencies $\{g_i, g_j\}$ is found, assign it to $\{i, j\}$ and mark all links in $I_{g_i g_j}^{ij}$ as unassigned, otherwise mark i and j as unassigned. Let $P := P \backslash \{i, j\}$.

4. Apply algorithm A to links that are left unassigned.

Unfortunately, the feasibility of solutions obtained in this way from a feasible solution cannot be guaranteed. Hence, in developing a local search technique based on this neighborhood, we also have to make sure that our solutions have a low degree of infeasibility. We measure this degree by the number of links in a solution that cannot be assigned a feasible frequency. If this number becomes too high, we apply one of the methods for the first order problem to restore feasibility of the current solution. We stop if our algorithm for the first order problem fails to produce a feasible solution.

4.3 Taboo search

We incorporate the neighborhood described in the previous section in the taboo search framework. Two criteria are used to select the best solution in the neighborhood at each step. First, we consider the neighbors with a minimum number of unassigned links. Second, among these solutions the one using a minimum number of distinct frequencies is selected.

Taboo search

1. Start with a feasible complete solution S. Let the taboo list T be empty. N and I are given integer numbers, denoting the size of the taboo list and the maximum number of iterations respectively. Set the counter i to 0. Let U be the maximum number of unassigned links allowed during the search.

2. Determine P - the set of neighbors of S for all seed frequencies taken from the set of used frequencies not in T.

3. Select a complete solution S_1 from P with a minimum number of used frequencies. If no such solution exists, select an incomplete solution S_1 from P with a minimum number of unassigned links and, in case of ties, with a minimum number of used frequencies.

4. If the number of unassigned links in S_1 is bigger than U, then apply the simulated annealing method for the first order problem, initiated with S_1. Let S be the solution obtained in this way. Go to step 6.

5. Include the seed frequency generating S_1 in T. If the size of T exceeds N, delete the earliest inclusion in T. Denote the solution S_1 by S.

6. Increase i by 1. If $i > I$, then stop, otherwise return to step 2.

4.4 Simulated annealing

We use the same neighborhood to define a simulated annealing method. Here a random solution from a neighborhood is selected and a specific cost function of this solution is evaluated. We define the cost function as the sum of the number of used frequencies and a hundred times the number of unassigned links. If we denote by C_o and C_n the cost of the old and the new solution respectively, then the search is continued with a new solution with probability $\min\{1, \exp(-(C_n - C_o)/T\}$, where T is a control parameter that decreases linearly every k iterations. Let T_0 and T_f be an initial and a final value of control parameter, respectively, and let δ be the decrement rate.

Simulated annealing

1. Start with a feasible complete solution S. Set the counter i to 0 and the control parameter T to T_0. Let U be the maximum number of unassigned links allowed during the search.

2. Select randomly a frequency used in S, and find a solution S_1 generated from S with this seed frequency.

3. Select a real number p from the uniform distribution on [0,1]. If p is larger than $\min\{1, \exp(-(C_n - C_o/T)\}$, then go to step 6.

4. If the number of unassigned links in S_1 is bigger than U, then apply the simulated annealing method for the first order problem, initiated with S_1. Let S be the solution obtained in this way. Go to step 6.

5. Denote the solution S_1 by S.

6. Increase i by 1. If $i > K$, then set T to σT and i to 0. If $T < T_f$, then stop, otherwise return to step 2.

4.5 Variable-depth search

The idea behind the variable-depth search procedure for the second order problem can be traced back to that of the neighborhood used in taboo search and simulated annealing, which is to eliminate one of the frequencies used in some given solution. However, this time it is done with a more exhaustive search. Namely, for each link i with a prohibited frequency in a given solution, frequencies of links in some connected component of the interference graph can be reassigned in search for improvement.

Variable-depth search

1. Start with a feasible complete solution S. Let $I_{\max}$ be the maximum number of iterations. Set the counter i to 0.

2. Choose randomly some *seed* frequency f used in the solution S. Prohibit the use of f for the assignment.

3. Determine a set P of all unassigned links and links with a frequency f in solution S. Mark all links in P as unassigned.

4. Select link k from P and its parallel link l. Let $P := P \backslash \{k, l\}$. Let $D = \{k, l\}$. Remove all labels.

5. Select link i from D and its parallel link j. Let $D := D \backslash \{i, j\}$. Find a pair of frequencies $\{g_i, g_j\}$, $g_i \neq f$, $g_j \neq f$, $|g_i - g_j| = d_{ij}$, such that g_i and g_j do not interfere with any labeled link, and such that

$$\{g_i, g_j\} = \arg \min_{h \in M_i, k \in M_j} |I^{ij}_{hk}|;$$

if ties occur, choose the pair $\{g_i, g_j\}$ that maximizes the minimum number of alternative frequencies at interfering links

$$\min_{l \in I^{ij}_{g_i g_j}} |F_l|.$$

If such a pair of frequencies $\{g_i, g_j\}$ is found, assign it to $\{i, j\}$ and mark all links in $I^{ij}_{g_i g_j}$ as unassigned, otherwise mark $\{i, j\}$ as unassigned and go to step 6. Assign a label to i and j. Add the set $I^{ij}_{g_i g_j}$ to D.

6. If D is not empty, return to step 5.

7. Remove all links with an assigned frequency from P. If P is not empty, return to step 4.

8. Apply algorithm A to unassigned links. If the current solution is complete and uses fewer frequencies than S denote the current solution by S. Increase i by 1. If $i > I_{\max}$, then stop, otherwise return to step 3.

5 Lower bounds

Our work was mainly concerned with the development of local search techniques for the given problem. These methods perform very well in practice and provide near-optimal solutions in a reasonable time. However, all these methods are based on intuitive considerations, which cannot be validated analytically. In this case virtually the only possible way to estimate objectively the quality of a local search approach is to compare its result with some lower bound. We develop here several lower bound techniques.

5.1 The first order problem

We will formulate a nonlinear optimization problem. An optimal solution to this problem will be a lower bound for the first order problem. We will describe a preprocessing technique and an enumeration scheme, which can be used to solve this problem under some special conditions. These special conditions are the following:

- A significant number of links have a preassigned frequency and at least some of these frequencies cannot be changed.

- The mobility coefficients M_i are not much smaller than the interference coefficients C_{ij}.

Consider the following decision variables:

$$x_i = \begin{cases} 1, & \text{if link } i \text{ is set to its preassigned frequency;} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly these variables are fixed to 1 for all links whose preassigned frequency cannot be changed. Note that even if a link does not have any preassigned frequency, we can specify one and set its mobility coefficient to zero without loss of generality. Denote by K_{ij} the cost of interference between i and j if both of them are set to their preassigned frequencies:

$$K_{ij} = \begin{cases} C_{ij}, & \text{if } |p_i - p_j| \leq d_{ij} \ ; \\ 0, & \text{otherwise.} \end{cases}$$

We denote by K_{ifj} the cost of interference between i and j if frequency f is assigned to i and j is set to its preassigned frequency:

$$K_{ifj} = \begin{cases} C_{ij}, & \text{if } |f - p_j| \leq d_{ij} \ ; \\ 0, & \text{otherwise.} \end{cases}$$

Now the problem can be formulated in the following way:

$$\min \sum_{i,j \in L, i<j} K_{ij} x_i x_j + \sum_{i \in L} M_i (1 - x_i) + \sum_{i \in L} (1 - x_i) (\min_{f \in D_i \setminus p_i} \sum_{j \in L} K_{ifj} x_j), \tag{5}$$

where $x_i \in \{0, 1\} \quad \forall i \in L$.

We will interpret this objective function for some link i. If the decision is made not to use the preassigned frequency for i, in other words if x_i is zero, then the second and the third terms in (5) occur for i. The second term is then simply a mobility cost of i. The third term in this case is the minimal cost of interference that may occur between i and all other links, which are set to their preassigned frequencies. Suppose now that the frequency of i is fixed to its preassigned value ($x_i = 1$). The cost of interference between i and all other links fixed to their preassigned frequencies is given by the first term. The variable x_i is also considered in the third term, determining the cost of the cheapest alternative frequency for links whose frequencies are different from their preassigned ones.

The only interference cost that is not covered by this objective function concerns the interference between links with frequencies different from their preassigned ones. Note that all cost coefficients in this problem are nonnegative.

Preprocessing In order to solve problem (5) we first try to reduce its size. For this purpose we will apply some preprocessing, that is, we fix some variables x_i to 1 or to 0 and prove that there is an optimal solution in which they will have these values. Define by F_0 a set of all indices (links) i for which x_i has been fixed to 0. In the same manner we define F_1 and F_2 for the free variables and the variables fixed to 1, respectively.

Suppose we want to fix x_i at 1. Consider some arbitrary solution in which $x_i = 0$. If we are able to show that the objective function of this solution cannot increase when x_i is fixed to 1, we are free to do so. A similar consideration holds for fixing x_i at 0.

We will now derive a sufficient condition for fixing variable x_i to 1 for some link i from F_1. Consider an arbitrary solution where $x_i = 0$. If we change x_i to 1 in this solution, then some cost terms may increase and new ones may appear, on the other hand a part of the objective function will decrease. We call the former phenomenon the *cost* of the change and the latter its *saving*. To be able to fix a variable, we would like to show that the saving of the change is not smaller than its cost.

The saving obtained by changing x_i from 0 to 1 is equal to

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in L \setminus i} K_{ifj} x_j. \tag{6}$$

In words, it consists of a mobility cost of i and an interference cost of the cheapest alternative to a preassigned frequency. We cannot evaluate this expression directly, because it contains decision variables. The principle remains valid, however, if we consider an upper bound on the cost of changing and a lower bound on the saving. Similarly, the cost of such a change is given by

$$\sum_{j \in L \setminus i} K_{ij} x_j + \sum_{j \in L} (1 - x_j) T_{ij}, \tag{7}$$

where

$$\begin{aligned} T_{ij} &= \min_{f \in D_j \setminus p_j} \Big(\sum_{l \in L \setminus i} K_{jfl} x_l + K_{jfi} \Big) - \min_{f \in D_j \setminus p_j} \Big(\sum_{l \in L \setminus i} K_{jfl} x_l \Big) && (8) \\ &\leq \min_{f \in D_j \setminus p_j} \Big(\sum_{l \in F_2 \cup F_1} K_{jfl} \Big) - \min_{f \in D_j \setminus p_j} \Big(\sum_{l \in F_2} K_{jfl} \Big) && (9) \\ &=: \delta_j(F_1, F_2) \ . \end{aligned}$$

At the same time

$$T_{ij} \leq \max_{f \in D_j \setminus p_j} K_{jfi} \ . \tag{10}$$

So we obtain the following upper bound for the expression (8):

$$T_{ij} \leq \min\{\delta_j(F_1, F_2), \max_{f \in D_j \setminus p_j} K_{jfi}\} =: \delta_{ij} \ . \tag{11}$$

The value of δ_{ij} can be calculated for every j. Let us now consider the difference between saving and cost. This value is at least

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in F_1 \cup F_2 \setminus i} K_{ifj} x_j - \sum_{j \in F_1 \cup F_2 \setminus i} K_{ij} x_j - \sum_{j \in F_0 \cup F_1 \setminus i} (1 - x_j) \delta_{ij} = \tag{12}$$

$$M_i + \min_{f \in D_i \setminus p_i} [\sum_{j \in F_1 \setminus i} ((K_{ifj} - K_{ij})x_j - (1 - x_j)\delta_{ij}) + \sum_{j \in F_2} (K_{ifj} - K_{ij})] + \sum_{j \in F_0} (-\delta_{ij}) \quad (13)$$

$$\geq M_i + \min_{f \in D_i \setminus p_i} [\sum_{j \in F_1 \setminus i} \min\{K_{ifj} - K_{ij}, -\delta_{ij}\} + \sum_{j \in F_2} (K_{ifj} - K_{ij}) + \sum_{j \in F_0} (-\delta_{ij})] \; . \quad (14)$$

Now a sufficient condition for fixing variable x_i at 1 is that

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in L \setminus i} \gamma_{ifj} \;\; \geq \;\; 0 \; , \quad (15)$$

$$\text{where} \quad \gamma_{ifj} = \begin{cases} -\delta_{ij}, & j \in F_0; \\ K_{ifj} - K_{ij}, & j \in F_2; \\ \min\{K_{ifj} - K_{ij}, -\delta_{ij}\}, & j \in F_1. \end{cases}$$

We will repeat this scheme once again to derive a sufficient condition for fixing x_i to 0. Consider some arbitrary solution where x_i is 1. We change the value of x_i in this solution to 0. The cost of such a change and its saving are now given by expressions (6) and (7), respectively. We now obtain a lower bound on (8):

$$\min_{f \in D_j \setminus p_j} (\sum_{l \in L \setminus i} K_{jfl} x_l + K_{jfi}) - \min_{f \in D_j \setminus p_j} \sum_{l \in L \setminus i} K_{jfl} x_l \geq \min_{f \in D_j \setminus p_j} K_{jki} =: \epsilon_{ij}. \quad (16)$$

Consider the difference between cost and saving of changing x_i from 0 to 1:

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in F_1 \cup F_2 \setminus i} K_{ifj} x_j - \sum_{j \in F_1 \cup F_2 \setminus i} K_{ij} x_j - \sum_{j \in F_0 \cup F_1 \setminus i} (1 - x_j)\epsilon_{ij} \leq \quad (17)$$

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in L \setminus i} \beta_{ifj} \; , \quad (18)$$

$$\text{where } \beta_{ifj} = \begin{cases} -\epsilon_{ij}, & j \in F_0; \\ K_{ifj} - K_{ij}, & j \in F_2; \\ \max\{K_{ifj} - K_{ij}, -\epsilon_{ij}\}, & j \in F_1. \end{cases}$$

A sufficient condition for fixing x_i at 0 is that

$$M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in L \setminus i} \beta_{ifj} \;\; \leq \;\; 0. \quad (19)$$

If the sufficient condition for fixing x_i at 1 or 0 is satisfied, we fix this variable, delete index i from the set F_1 and include it in F_2 (respectively F_0). Note that once the set F_2 or F_0 is modified, the sufficient conditions for preprocessing of all $free$ variables (x_j for all $j \in F_1$) are changed, in fact they are strengthened. So, we repeat this preprocessing cyclicly until no free variable can be fixed anymore after the last successful preprocessing.

In practice, the specific structure of an instance determines how many variables can be fixed by preprocessing. The favorable conditions for that have been loosely defined in the beginning of this section. For some of the CELAR test instances, we were able to solve the lower bound problem completely by preprocessing, for others we had to solve relatively small subproblems left after preprocessing by enumeration.

Efficient enumeration We will now describe the procedure to perform this enumeration efficiently. Starting with an initial solution this procedure uses the so-called Gray code to change one variable at a time. The number of times the variable has to be swapped is however different for all variables. We will use this fact to swap more frequently those variables that require less computational work for updating the cost function.

We will first describe how the objective function can be updated. When the preprocessing stage is completed, we can calculate for each link i in F_1 the cost of interference of preassigned frequency of i with all links in F_2. Denote it by C_{iF_2}. Define $I(i)$ as the set of all links adjacent to i in the interference graph. Suppose all variables have been given a value and the objective function of this solution has been calculated. We also calculate and store some auxiliary values:

$$V_i \;=\; \min_{f \in D_i \setminus p_i} \sum_{j \in I(i) \cap (F_1 \cup F_2)} K_{ifj} x_j \quad \forall\, i \in F_1 \cup F_0 \text{ for which } x_i = 0.$$

Consider now the change of some variable $x_i,\ i \in F_1$, from 1 to 0. The objective function of a new solution is computed by adding the following value to the objective function of the previous solution:

$$\begin{aligned} &- \sum_{j \in F_1 \cap I(i)} K_{ij} x_j - C_{iF_2} + M_i + \min_{f \in D_i \setminus p_i} \sum_{j \in I(i) \cap (F_1 \cup F_2)} K_{ifj} x_j \\ &- \sum_{l \in I(i) \cap (F_1 \cup F_0)} (1 - x_l)[V_l - \min_{f \in D_l \setminus p_l} \sum_{j \in I(l) \cap (F_1 \cup F_2)} K_{lfj} x_j]. \end{aligned} \tag{20}$$

The value V_i, which is the fourth term in this expression, is stored. Now once x_i has been set to 0, the values V_j have to be recalculated for all $j \in I(i) \cap (F_1 \cup F_0)$ for which $x_j = 0$.

Let us consider the opposite change of some variable $x_i,\ i \in F_1$, from 0 to 1. The increment of the objective function is given in this case by the following expression:

$$\sum_{j \in F_1 \cap I(i)} K_{ij} x_j + C_{iF_2} - M_i - V_i + \sum_{l \in I(i) \cap (F_1 \cup F_0)} (1 - x_l)[\min_{f \in D_l \setminus p_l} (\sum_{j \in I(l) \cap (F_1 \cup F_2)} K_{lfj} x_j + K_{lfi}) - V_l]\ .$$

Here again, after setting x_i to 1, values V_j have to be recalculated for all $j \in I(i) \cap (F_1 \cup F_0)$ for which $x_j = 0$.

The amount of computation for updating the cost function can be estimated for each link i in F_1 as

$$|I(i) \cap F_1| + |I(i) \cap (F_2 \cup F_1)|(|D_i| - 1) + \sum_{j \in I(i) \cap (F_1 \cup F_0)} |I(j) \cap (F_2 \cup F_1)|(|D_j| - 1)\ .$$

We enumerate the links in F_1 from 0 to $|F_1| - 1$ in nondecreasing order of this estimate. In order to determine the next link to be swapped we use the auxiliary array `Toggle` of $|F_1|$ elements from $\{0, 1\}$. This array is initiated with all ones. At each step we swap the first i elements of this array, where i is the first element with zero entry after swapping. Index i determines now the new link in F_1 to be swapped. This operation is performed until all entries of `Toggle` are zero.

We incorporated this lower bounding technique in a branch and bound framework. We use the best solution obtained by the approximation algorithms for the first order problem as an upper bound. Branching is performed by splitting the domain of some link into two or more parts. If for some branch in the first level its lower bound value is strictly higher than the value of an upper bound, we conclude that the corresponding subset of the split domain is not used in any optimal solution and can thus be dropped. Using this argument for the CELAR instances we were able to reduce the domains of the links significantly, even though we did not use the full branching scheme for reasons of computation time and considered splitting domains of at most two links at a time.

5.2 The second order problem

One possible lower bound for the number of used frequencies of the RLFAP is the size of the largest clique in its interference graph. We will focus on this lower bound. In order to determine the largest clique we develop a branch and bound algorithm. We define variables x_i in the following way:

$$x_i = \begin{cases} 0, & \text{if link i is excluded from the clique;} \\ 1, & \text{if link i is included to the clique;} \\ 2, & \text{if this decision is yet to be made.} \end{cases}$$

We define G to be a subgraph of an interference graph induced on all links i for which x_i is nonzero. The upper bound is the minimum degree in G of links i for which $x_i = 1$. We also make sure that all such links form a clique. We obtain a lower bound by the following greedy heuristic.

Algorithm MC

1. Let S be a set of all links i, whose variable x_i is equal to 2. The graph G' is equal to G.

2. If G' is a clique, then stop.

3. Select a link from S of smallest degree in G' and remove it from S and G'. Ties are broken randomly. Return to step 2.

If during branching a larger clique is found, we terminate the branching process, set all variables to 2 and eliminate from G all links with degree smaller than the size of the largest clique. Initially let G be interference graph defined on a ground set L. We describe our branch and bound algorithm recursively:

Initial step

1. All variables x_i are set to 2. If the current size C of the maximal clique is zero, apply algorithm **MC** to initiate it.

2. Delete from G all nodes with degree smaller than C and repeat this until all nodes of G have a degree not smaller than C.

3. Arrange the nodes of G in nonincreasing order of their degree. Let the current clique K be empty.

Recursive step

1. Select the first node i of G with $x_i = 2$. If no such i is found, backtrack to the previous active node. Stop if there is no active node.

2. If $K \cup i$ is a clique, then let $K = K \cup i$ and set x_i to 1, else go to step 5.

3. If $|K| = C$, apply algorithm **MC** to find a clique. If the size of a clique found by **MC** is larger than C, then set C to this value and return to Initial step, otherwise go to step 5.

4. Let i be an active node. Perform the recursive step.

5. Set x_i to 0. If the smallest degree of nodes in K is not smaller than C, perform the recursive step. Backtrack to the previous active node. Stop if there is no active node.

Before we actually apply this algorithm to find a lower bound, we use some simple implication arguments to introduce extra constraints on an instance. These arguments are based on two special properties of all test instances, namely that all d_{ij} in constraints (2) are equal, and that, if the frequency of a link is fixed, then the choice of the frequency for its parallel link is unique. So, for each constraint of type (1) between links i and j there should be a constraint between their parallel links, and if some link interferes with a pair of parallel links, so should its parallel link.

Having a tool to obtain a largest clique we decided to strengthen further the lower bound on the RLFAP. Consider some clique K in the interference graph, and let K' be a clique of the same size as K such that a difference between the sets of nodes of K and K' is a singleton. Then there are only two non-interfering nodes i and j in the union of these sets that can be either connected or contracted [2]. Note that if i and j are connected, we immediately obtain a clique larger than K. Now suppose K is the maximal clique in the interference graph and we want to prove that we need at least one extra frequency, besides those used in K, to obtain a feasible solution to the RLFAP. If we are able to find K' as described above, we only have to consider the contraction procedure. So, considering this as a branching rule we are only left to check a path in the decision tree to improve our lower bound.

The lower bound for the second order problem of the RLFAP is, apart from the pre-processing step, merely based on the pairwise interference of links. One possible way to take the other problem data into account is to check the existence of a feasible frequency assignment to a clique after a contraction procedure has been performed. Indeed, if we are able to prove that there is no feasible assignment after contracting two nodes as described

in the previous paragraph, we immediately improve our lower bound. Note that after contraction of two nodes i and j in a new node l the interference distances of l are defined as

$$d_{lm} = \max\{d_{im}, d_{jm}\} \quad \forall m \in I(i) \cup I(j) \ .$$

The problem we consider is the following. Given a complete graph K and an interference distance function defined on the edges, does there exist an assignment of frequencies from domain D to the nodes, so that the distance between the frequencies assigned to any pair of nodes will be at least their interference distance. We call it the *leaf* problem. Before we prove the NP-completeness of this problem we want to recall that, although the maximal clique problem is also NP-hard, we wanted to test the possibility of embedding an exact algorithm for the leaf problem into another algorithm with exponential running time.

Lemma 2 *The leaf problem is NP-complete.*

Proof We will construct a reduction from the Hamiltonian path problem (HPP). An instance of HPP is an undirected graph $G = (V, E)$ and the question: 'is there a Hamiltonian path in this graph?' Let domain D be the set of consecutive integers from 1 to $|V|$. Define an instance of the leaf problem as the complete graph on the set V with domain D and an interference distance d_{ij} is 1 if there is an edge between i and j in E and 2 otherwise. We argue that if there exists a feasible solution to the instance of the leaf problem constructed in this way, then there is a Hamiltonian path in G, and vice versa. Indeed, consider a feasible solution of the leaf problem and a permutation of the nodes in increasing order of frequencies assigned to them in this solution. There is an edge in E between any two nodes adjacent in this permutation and each node from V appears exactly once , so it is a Hamiltonian path in G. Obviously, when a Hamiltonian path exists, one can assign frequency i to the i'th node in the path and thus create a feasible frequency assignment. □

6 Results

We tested our methods on eleven CELAR instances. These instances are available via an anonymous ftp at `ftp.win.tue.nl` in the directory `/pub/techreports/CALMA/Instances`. The computational results are summarized in the Table 1. The instances labeled with * have an optimal solution of value 0 in the first order problem and the entries in the table for these instances are the objective value of the second order problem. The computation times of the solution methods for first order problem for those instances are no longer than one minute; in fact, they are less than two seconds for instances 1, 2 and 3. Computation times are given in seconds of CPU time on a SUN SPARC 4 workstation. The lower bounds we obtained for instances 9 and 10 are the only nontrivial ones known for the first order problem. The approximation algorithms for the second order problem perform equally well on these test instances. We attribute this performance mainly to the neighborhood

Table 1: Computation results for real life instances

inst.	size		upper bound						lower bounds
	var.	constr.	taboo search		variable-depth		sim. annealing		
			value	*time*	value	*time*	value	*time*	
1*	916	5548	16	*705*	16	*1919*	16	*396*	14
2*	200	1235	14	*7*	14	*1*	14	*2*	14
3*	400	2760	14	*60*	14	*48*	14	*72*	14
4*	680	3967	46	*166*	46	*31*	46	*80*	46
5*	400	2598					792	*49*	792
6	200	1322			3532	*456*	3671	*65327*	
7	400	2865			344103	*8363*	567949	*26569*	
8	916	5744			299	*16534*	276	*716*	
9	680	4103			15667	*21*	15665	*372*	14969
10	680	4103			32456	*210*	32456	*24*	32144
11*	680	4103	22	*735*	24	*10*	24	*223*	22

inst.	instance number
size var.	the number of links in the instance
size constr.	the number of constraints
*	instance of the second order problem

structure. The taboo search framework performs better in case the feasible solutions are sparse, as in instance 11.

Our neighborhood for the first order problem happens to be less flexible than the one for the second order problem. It is effective in the simulated annealing framework, but we were not able to obtain any satisfactory performance of the taboo search method based on this neighborhood. The difficulty of simulated annealing with instances 6 and 7 can be explained by a fundamental problem of this method. Namely, the conventional cooling schemes do not work smoothly when the cost coefficients of the objective function are of different orders of magnitude, which is the case for these interferences. But the performance of simulated annealing on the instances where this problem does not occur is robust, and we used this method as a subroutine for approximation methods for the second order problem, as described in section 4.

We have also applied our algorithms to ten random instances due to H.P. van Benthem available at `ftp.dutiosd.twi.tudelft.nl` in the directory `/pub/others/rlfap`. The results are listed in the following the Table 2 in the same format as the Table 1, except for the best known solutions that are listed in place of the lower bounds. One of the aims of our work was to show that local search techniques are very flexible and can be easily modified to obtain approximate solutions for related problems. For this we considered a different variant of the frequency assignment problem which also stems from a practical application. In this problem we are given a set of forces and a set of possible locations for each force. Radio frequencies assigned to the forces at a same location have to satisfy

Table 2: Computation results for random instances

inst.	size		upper bound						best known
	var.	constr.	taboo search		variable-depth		sim. annealing		
			value	*time*	value	*time*	value	*time*	
1*	200	1171	14	*27*	14	*22*	14	*36*	12
2*	200	1143	46	*49*	46	*2*	46	*32*	46
3*	200	1160	18	*289*	18	*6*	18	*71*	18
4*	200	1143	20	*101*	20	*80*	20	*57*	20
5	200	1125			5	*210*	6	*146*	4
6*	916	5177	38	*260*	42	*286*	40	*321*	
7*	916	5173	48	*11016*	48	*11016*	48	*11016*	48
8*	916	5262	40	*130*	40	*314*	40	*463*	40
9	916	5183			32	*6380*	13	*10765*	
10	916	5123			456	*6915*	407	*11435*	

inst.	instance number
size var.	the number of links in the instance
size constr.	the number of constraints
*	instance of the second order problem

certain complicated interference constraints. Several interference levels are acceptable at each location corresponding to different parameter settings in the interference constraints. Each interference level has a penalty associated with it. For each force we are asked to select a frequency from a given domain so that:

- for each possible location of a force an acceptable interference level is achieved;
- the sum of penalties over all locations is minimized.

We were able to apply the simulated annealing algorithm for the first order problem with modified data structures to obtain approximate solutions to real-life instances of this problem.

7 Conclusions

Our study showed that local search is a powerful tool to solve various frequency assignment problems. However, for an efficient implementation of this method the special structure of a problem has to be used to design an appropriate neighborhood function. We observed the following advantages of the local search approach for real-life hard combinatorial problems:

- local search is easy to implement and to tune for a particular problem at hand;
- it is able to generate good solutions in limited time;

- it requires only modest computer facilities.

In order to guarantee quality of approximate solutions obtained by a local search a lower bounding technique has to be developed separately.

Acknowledgments

This work has been performed in the framework of the EUCLID RTP 6.4 CALMA project [6]. We had an extensive exchange of ideas with other participants during this project.

We are especially grateful to Jan Karel Lenstra for his involvement and numerous useful comments on the contents of this paper.

References

[1] E.H.L. Aarts and J.H.M Korst, Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural networks, Wiley, Chichester, 1989.

[2] C. Berge, Graphs and hypergraphs, North-Holland, Amsterdam, 1976.

[3] M.R Garey and D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, Freeman, San Fransisco, 1979.

[4] F. Glover, Tabu search - Part I, ORSA Journal on Computing 1, 1989, pp. 190-206.

[5] F. Glover, Tabu search - Part II, ORSA Journal on Computing 2, 1990, pp. 4-32.

[6] W. Hajema, M. Minoux, C. West, Statement of the radio link frequency assignment problem, Request for proposals on the CALMA project, Technical appendix, 1993.

[7] T.A. Lanfear, Graph theory and radio frequency assignment, Allied Radio Frequency Agency, NATO Headquarters, Brussels, 1989.

[8] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operations Research 21, 1973, pp. 498-511.